INTRODUCTION TO DEEP LEARNING

CSE 574 - Introduction to Machine Learning (Fall 2017)

PARITOSH WALVEKAR
pwalveka@buffalo.edu
Person Number - 50248710

KETAN SHAH
ketansha@buffalo.edu
Person Number - 50247131

BHAVIK GALA
bhavikna@buffalo.edu
Person Number - 50248608

CONTENTS

Brief

The project aims at implementation of a convolutional neural network on the celebA dataset to identify whether the celebrities have glasses on them.

The dataset provided contains two hundred thousand images of celebrities either with the glasses or without the glasses.

The model is built using a public library called tensorflow. Tensorflow exposes important functions used in building a convolutional neural network with five layers including two convolution layers (which comprises of convolution and pooling), one hidden and one output layer.

This project is helps give an introduction to deep learning methods.

Implementation of CNN

We have implemented the convolutional neural network with five layers which involve one convolutional layer, one polling layer, one combined layer (involving convolution and pooling in the same layer), dense layer and logins layer.

A function is written which will divided the celebA dataset into training(80%) and test(20%) datasets. Dropout regularization is implemented using the tensorflow function *tf.layers.dropout*.

The loss for the evaluation of the neural network has been calculated using the one hot vector concept.

The module *glassesOrNoGlasses.py* runs the model for four times on initial dataset, a bigger dataset, augmented dataset and on the dataset with a higher resolution. It also prints out the performance evaluation of the model in all the four cases.

Train Model

We first trained our model using a training set of 50000 images from the celebA list dataset with resolution 50x50 (50 pixels in height and 50 pixels in width). We used the tensorflow CNN code for this. We have extracted the features from the image using first a convolution layer followed by a pooling layer. This is again followed by a convolution layer and a pooling layer.

Finally the out from the second pooling layer is passed to 2 dense layers. The second dense layer gives the output.

Our basis for training is to predict if the person in the inout image is wearing spectacles or not. The error is minimized using the back propagation method.

Tuning Hyperparameters

As far as tuning the hyper parameters are concerned, we did a bit less tuning as compared to the previous three projects. When the resolution of the image was tweaked to contain more pixels, we improved the accuracy by 0.01%

The above result imply that given a large amount of compute power at disposal, we can use high resolution images to train the model which in the interim should improve the accuracy drastically.

Retraining using higher resolution

We tried training the model using a higher resolution. For this we increased the resolution to 100 by 100. This only increased the accuracy by 0.3 %. The reason for this might be that the number of total images containing the people wearing spectacles is very less. We also found that many images were wrongly tagged. For example some images where the person does have spectacles on, the output of that image says otherwise. These reasons might be the reason for not being able to improve the accuracy of the model.

Bigger Training Set

In an effort to try achieving better accuracy we increased the size of the training set from 50000 to 150000. Again the accuracy just increased by 0.1 %. The reason again for this might be lesser no. of spectacle images in the whole training dataset. Out of 200000 images only about 9000 images had people with spectacles on.

Data Augmentation

We tried increasing the data set by adding more images with people having spectacles by generating more images out of them by applying various transformations like random rotating, scaling, randomly changing the hue and saturation of the images. By doing this the accuracy again increase by very small amount that is 0.25 %.

Code Execution Manual

What's included in the zip?

1. glassesOrNoGlasses.py
2. helpers
3. model_50 (for 50x50 resolution images)
4. model100 (for 100x100 resolution images)
5. testImageAugmentation.py

How to run the code?

1. Unzip the file titled *proj4code.zip.*
2. On the root level of the folder, execute the following command

```
python3 glassesOrNoGlasses.py
```

Environment Details

1. Python v3.6
2. Mac OS v10.12
3. Conda v4.3.25
4. Numpy v1.13.1
5. Tensorflow v1.1.0

Console Output

First run evaluation: training set: 50000 samples
INFO:tensorflow:Restoring parameters from /model/model.ckpt-3616
INFO:tensorflow:Finished evaluation at 2017-12-04-03:41:30
INFO:tensorflow:Saving dict for global step 3616: accuracy = 0.953159, global_step = 3616, loss = 0.142733
{'accuracy':0.95315892, 'loss':0.14273272, 'global_step': 3616}

Evaluation with bigger training set: training set: 150000 samples
INFO:tensorflow:Starting evaluation at 2017-12-04-20:36:17
INFO:tensorflow:Restoring parameters from ./model_50/model.ckpt-2500
INFO:tensorflow:Finished evaluation at 2017-12-04-20:37:01
INFO:tensorflow:Saving dict for global step 3616: accuracy = 0.9551568, global_step = 2500, loss = 0.135721
{'accuracy':0.95515685, 'loss':0.13572120, 'global_step': 2500}

Data augmentation: resolution 50 x 50
INFO:tensorflow:Starting evaluation at 2017-12-06-05:36:17
INFO:tensorflow:Restoring parameters from ./model_50/model.ckpt-2500
INFO:tensorflow:Finished evaluation at 2017-12-06-05:36:56
INFO:tensorflow:Saving dict for global step 2500: accuracy = 0.958133, global_step = 2500, loss = 0.14736
{'accuracy': 0.95813326, 'loss': 0.14735997, 'global_step': 2500}

Resolution: 100 x 100
INFO:tensorflow:Starting evaluation at 2017-12-06-16:54:27
INFO:tensorflow:Restoring parameters from ./model100/model.ckpt-2500
INFO:tensorflow:Finished evaluation at 2017-12-06-16:57:04
INFO:tensorflow:Saving dict for global step 2500: accuracy = 0.957097, global_step = 2500, loss = 0.629282
{'accuracy': 0.95709673, 'loss': 0.629282, 'global_step': 2500}